Claims

What is claimed is:

1. A serialization and deserialization architecture, comprising:

a serialization and deserialization portion adapted to provide serialization information on a data structure;

a pluggable formatter adapted to receive the serialization information and provide a serialized stream in an externalized format defined by the pluggable formatter; and

a utility and contract portion adapted to maintain the invariants with serializing and deserializing the data structure.

2. The architecture of claim 1, the data structure being an object

3. The architecture of claim 1, the data structure being a graph of objects.

4. The architecture of claim 1, the serialization and deserialization portion employing a rule set to define the serialization information to be provided to the pluggable formatter.

5. The architecture of claim 4, the rule set being provided in the data structure.

6. The architecture of claim 4, the rule set being provided in a third party file.

7. The architecture of claim 4, the rule set specifying a different data structure to be serialized.

8. The architecture of claim 7, the different data structure being a reference data structure type containing information about the data structure.

9. The architecture of claim 7, the different data structure being a remote object type.

10. The architecture of claim 1, the pluggable formatter adapted to receive a serialized stream of a data structure and decode the serialized stream and the utility and contract portion providing a data structure type for deserializing the data from the serialized stream, the pluggable formatter and the utility and contract portion working in conjunction to populate the data structure type with data from the serialized stream utilizing the serialization information.

11. The architecture of claim 10, the utility and contract portion further adapted to track data structures and provide fixups due to forward references.

12. The architecture of claim 10, the utility and contract portion further adapted to swap a reference data structure type with an actual data structure.

13. The architecture of claim 1, the externalized format being XML.

14. The architecture of claim 1, the externalized format being binary.

15. A system for serializing a graph of objects into a serial stream, comprising:

a plurality of rule sets defining serialization information about object types;

a serialization selector determining a rule set of the plurality of rule sets to utilize in providing serialization information to a formatter, the plurality of rule sets comprising a first rule set that allows a user to define serialization

information to be provided in the object itself and a second rule set that allows a user to define serialization information for the object in a third party object.

16 The system of claim 15, the first rule set allowing a user to provide markings in the object, such that the first rule set provides default serialization information about that object using data in the object.

17. The system of claim 15, the first rule set allowing a user to restrict the serialization information provided to the formatter.

18. The system of claim 15, at least one of the first rule set and the second rule set allowing a user to define a reference object to be serialized in place of the object.

19. The system of claim 18, the reference object containing information about the object.

20. The system of claim 18, the reference object being a marshal by reference object.

21. A system for deserializing a serial stream into a graph of objects, comprising:

a plurality of rule sets defining serialization information about object types;

a serialization selector adapted to determine a rule set of the plurality of rule sets on how an object was serialized, so that objects can be deserialized by a formatter;

an object type service adapted to create an unitialized instance of an object type for deserialization of an object;

an object tracking service adapted to track objects during the deserialization process and register fixups due to forward references; and

a fixup service adapted to fill in objects with data associated with the forward references.

22 The system of claim 21, further comprising an object swapping service adapted to extract serialization information from a reference object.

23 The system of claim 22, the object swapping service replacing the reference object with a proxy object.

24 The system of claim 22, the object swapping service replacing the reference object with an actual object relating to the serialization information extracted from the reference object.

25. An architecture for facilitating serialization of a graph of objects into a serial stream, comprising:

a formatter having a serialization decision loop and a pluggable formatter portion, the serialization decision loop retrieving an object from a graph of objects and determining serialization information to be provided to the pluggable formatter portion, the pluggable formatter portion providing a serialized stream populated with serialization information in an externalized format defined by the pluggable formatter portion;

a surrogate selector defining whether an object type has a surrogate that defines the serialization information to be provided to the formatter; and

a serializable interface implementable into a class type that allows a user to define the serialization information to be provided to the formatter within the class.

26. The architecture of claim 25, the serialization decision loop determining whether an object type has a surrogate selector and employing the surrogate selector to retrieve a surrogate that defines the serialization information to be provided to the pluggable formatter portion for objects of that type.

27. The architecture of claim 25, the serialization decision loop determining whether an object type employs the serializable interface and employing the methods of the serializable interface within the object to provide the pluggable formatter portion with serialization information for a given object of that type.

28. The architecture of claim 25, the serialization decision loop determining whether an object type includes serializable markings and employs a default serialization routine to provide the pluggable formatter portion with serialization information for a given object of that type.

29. The architecture of claim 25, further comprising an object reference interface implementable into a class, so that a user can define an object reference type to be serialized instead of objects of that type corresponding to the class.

30. The architecture of claim 29, the serializable interface allowing a user to redefine an object type to an object reference type within the class and reference a class implementing the object reference interface so that the object reference type can be serialized instead of objects of that type corresponding to the class.

31. The architecture of claim 25, further comprising a marshal by reference interface implementable into a class to redefine the class as a remote object.

32. The architecture of claim 29, further comprising a remote surrogate that provides a marshal reference object for objects that implement the marshal by reference interface.

33. The architecture of claim 25, the serialization decision loop employing an object ID generator to assign object IDs to each object in the graph of objects.

34. An architecture for facilitating deserialization of a serial stream into a graph of objects, comprising:

a plurality of rule sets defining serialization information about object types;

a serialization selector adapted to determine a rule set of the plurality of rule sets on how an object was serialized, so that objects can be deserialized by a formatter;

a serialization binder component adapted to specify an object type for deserialization of an object; and

an object manager adapted to manage the deserialization of object by the formatter.

35 The architecture of claim 34, further comprising a formatter services component adapted to register objects with the object manager as the objects are retrieved from the formatter and instantiate an object type based on the type specified by the serialization binder component.

36 The architecture of claim 35, the formatter services component being further adapted to extract serialization information from a reference object.

37 The architecture of claim 36, the object manager being further adapted to call a get real object component and insert the real object in place of the reference object.

38. The architecture of claim 36, the formatter services component replacing the reference object with a proxy object.

39. The architecture of claim 21, further comprising a callback interface implementable into a class, the object manager being adapted to perform additional fixups on the object graph after all of the objects in the object graph have been deserialized for objects implementing the callback interface.

40. A method for serialization of a graph of objects into a stream, the method comprising:

retrieving an object from the graph of objects;

assigning the object a unique identification number;

determining if a surrogate selector is associated with the object for objects of that type and employing the surrogate selector to access a surrogate defining serialization information about the object;

determining if the object employs a serializable interface if the object does not have a surrogate, and accessing serialization information within the object if the object employs the serializable interface and employing this serialization information to determine what portion of the object will be serialized;

determining if the object has serialization markings if the object does not employ a serializable interface or a surrogate and providing a default serialization routine for providing serialization information;

pushing the object to a data structure and populating the data structure based on the selected serialization information; and

serializing the data structure to an externalized format defined by a pluggable formatter.

41. The method of claim 40, the pushing the object to a data structure and populating the data structure based on the selected serialization information being repeated for each object in the graph of objects prior to serializing the data structure to an externalized format defined by a pluggable formatter.

42. A method for deserialization of a stream into a graph of objects, the method comprising:

receiving a decoded serialized stream from a pluggable formatter;

retrieving an object from the decoded serialized stream;

determining an object type for deserialization of the object;

instantiating an uninitialized instance of the object type;

selecting serialization information for the object from one of surrogate, a serializable interface employed in the object and a default serialization routine, the serialization information being employed in the deserialization process;

populating the uninitialized instance of the object type based on the selected serialization information;

registering fixups for the object during the deserialization process due to forward references within the object; and

performing fixups on the objects in the object graph caused by the forward references.

43. The method of claim 42, further comprising retrieving real objects for any object reference types that are provided in place of the real object and placing the real objects in the graph in place of the reference object type.

44. The method of claim 42, further comprising creating a proxy for any object reference type that is derived from a marshal by reference object.

45. A computer readable medium having computer executable components comprising:

a formatter programmed to serialize objects based on a rule set for an object type, the rule set being provided in either the object or a third party object;

an object manager programmed to manage the deserialization of objects by the formatter.

46. The computer readable medium of claim 45, the formatter having a decision loop portion programmed to retrieve an object and invoke a serialization selection routine, the serialization selection routine determining serialization information on the object, the serialization information being user definable in the object through one of methods defined in a serializable interface implemented in the class, methods defined in a surrogate class provide by a user and markings provided in the object invoking a default serialization format.

47. The computer readable medium of claim 45, the formatter having a pluggable formatter portion for providing the serialization information in an external format defined by the pluggable formatter portion.

48. The computer readable medium of claim 45, the pluggable formatter being programmed to receive a serialized stream and decode the serialized stream into a graph of objects.

49. The computer readable medium of claim 45, further comprising a formatter services component adapted to receive the serialized stream of the object and instantiate a given object type for deserialization of the serialized stream.

50. The computer readable medium of claim 49, further comprising a serialization binder component adapted to determine the given object type to instantiate.

51. The computer readable medium of claim 50, the object type being one of a similar type of the object being deserialized and a different type defined by a user.

52. The computer readable medium of claim 45, the object manager component being programmed to track forward references to additional objects of

the graph of objects down the stream and perform fixups to the objects upon receipt of the additional objects.

53. The computer readable medium of claim 45, the object manager retrieving a real object component for retrieving real objects for any object reference types provided in place of the real objects and placing the real object in the graph.

54. The computer readable medium of claim 45, the formatter services component being further adapted to create a proxy for any object reference that is derived from a marshal by reference object.

55. A system for serializing and deserializing a graph of objects, comprising:

means for providing serialization information of an object to a formatter, so that the formatter can serialize the object in a selectable externalized format, the means for providing serialization information being in either the object or a third party object;

means for tracking deserialization of the object outside the pluggable formatter; and

means for instantiating an uninitialized instance of a given object type, so that the object information can populate the object type.

56. The system of claim 55, further comprising means for performing fixups on the objects due to forward references.